IN THE CLAIMS

1. (Currently Amended) A computer-implemented method for a first component to invoke a second component asynchronously in an object-oriented computing environment, the computer-implemented method comprising:

receiving at an asynchronous proxy an asynchronous request from a first object-oriented component residing at a first server to invoke a second object-oriented component residing at a second server wherein the request has a void return type and is not associated with application-specific exceptions ~~wherein the first server is connected to the second server over a network interface~~;

setting an exception listener on the asynchronous proxy and a scope of the second component, the exception listener being registered for the second component;

~~maintaining the a scope of the received request at the second server;~~

storing the request and the scope in a queue on the asynchronous proxy;

providing a thread for identifying the received request and invoking the second component, wherein the thread identifies an exception listener object-oriented component for handling exceptions associated with the invocation of the second component, wherein the exception listener is registered on an asynchronous proxy, is stateless and is operable to handle a plurality of types of exceptions from a plurality of different components.

2. (Canceled) The computer-implemented method of claim 1, wherein the request has a return type of void.

3. (Canceled) The computer-implemented method of claim 1, wherein the request is associated with no application specific exceptions.

4 (Previously Amended) The computer-implemented method of claim 1, wherein the first and second components reside in environments allowing components to directly invoke other components.

5. (Original) The computer-implemented method of claim 1, wherein the first and second components are Enterprise Java Bean components.

6. (Original) The computer-implemented method of claim 5, wherein the first and second components are associated with a container.

7. (Previously Amended) The computer-implemented method of claim 6, further comprising placing the request from the first component in a queue.

8. (Previously Amended) The computer-implemented method of claim 7, wherein a worker thread dequeues the received request after receiving a transaction commit signal from the container.

9. (Original) The computer-implemented method of claim 8, wherein the exception listener receives the exception and the scope of the exception.

10. (Currently Amended) A computer-implemented method for a first object-oriented component to invoke a second object-oriented component asynchronously in an object-oriented environment, the computer-implemented method comprising:

transmitting a an asynchronous request from the first object-oriented component residing at a first server to invoke the second object-oriented component residing at a second server, the first and second object-oriented components operating in environments allowing direct invocation of the second component by the first component;

receiving the asynchronous request wherein the request has a void return type and is not associated with application-specific exceptions;

registering an exception listener object-oriented component on an asynchronous proxy and a scope of the second object-oriented components, the exception listener being registered for the second component and, the asynchronous proxy being associated with the second component, wherein the exception listener is stateless and is operable to handle a plurality of types of exceptions from a plurality of different components.

11. (Original) The computer-implemented method of claim 10, wherein the asynchronous proxy has the same type as the second component.

12. (Cancelled) The computer-implemented method of claim 10, wherein the request is associated with no application specific exceptions.

13. (Original) The computer-implemented method of claim 10, wherein the first and second components are associated with separate servers.

14. (Original) The computer-implemented method of claim 10, wherein the first and second components are Enterprise Java Bean components.

15. (Original) The computer-implemented method of claim 14, wherein the first and second components are associated with a container.

16. (Currently Amended) A computer program product comprising computer code for a first component to invoke a second component asynchronously, the computer program product ~~readable medium~~ comprising:

computer code for receiving at an asynchronous proxy an asynchronous request from a first object-oriented component residing at a first server to invoke a second object-oriented component residing at a second server, wherein the request has a void return type and is not associated with application-specific exceptions ~~wherein the first server is connected to the second server over a network interface~~;

computer code for setting an exception listener on the asynchronous proxy and a scope of the second component, the exception listener being registered for the second component;

~~computer code for maintaining the scope of the received request at the second server;~~

computer code for storing the request and the scope in a queue on the asynchronous proxy;

computer code for providing a thread for identifying the received request and invoking the second component, wherein the thread identifies an exception listener object-oriented component for handling exceptions associated with the invocation of the second component, wherein the exception listener is registered on an asynchronous proxy, is stateless and is operable to handle a plurality of types of exceptions from a plurality of different components.

a computer-readable medium for storing the computer codes.

17. (Original) The computer program product of claim 16, wherein the request has a return type of void.

18. (Original) The computer program product of claim 16, wherein the request is associated with no application specific exceptions.

19. (Previously Amended) The computer program product of claim 16, wherein the first and second components reside in environments allowing components to directly invoke other components.

wherein the first and second components are associated with separate servers.

20. (Original) The computer program product of claim 16, wherein the first and second components are Enterprise Java Bean components.

21. (Original) The computer program product of claim 20, wherein the first and second components are associated with a container.

22. (Original) The computer program product of claim 21, further comprising placing the request from the first component is placed in a queue.

23. (Original) The computer program product of claim 22, wherein the worker thread dequeues the received request after receiving a transaction commit signal from the container.

24. (Original) The computer program product of claim 23, wherein the exception listener receives the exception and the scope of the exception.

25. (Currently Amended) A computer program product for a first object-oriented component to invoke a second object-oriented component asynchronously in an object-oriented environment, the computer program product comprising:

computer code for transmitting a an asynchronous request from a first object-oriented component residing at the first server to invoke the second object-oriented component residing at a second server, the first and second object-oriented components operating in environments allowing direct invocation of the second component by the first component;

computer code for receiving the asynchronous request wherein the request has a void return type and is not associated with application-specific exceptions;

computer code for registering an exception listener object-oriented component on an asynchronous proxy, and for setting a scope associated with the second object-oriented components, the exception listener being registered for the second component and the asynchronous proxy associated with the second component, wherein the exception listener is stateless and is operable to handle a plurality of types of exceptions from a plurality of different components;

a computer-readable medium for storing the computer codes.

26. (Original) The computer program product of claim 25, wherein the asyenhronous proxy has the same type as the second component.

27. (Original) The computer program product of claim 25, wherein the request is associated with no application specific exceptions.

28. (Original) The computer program product of claim 25, wherein the first and second components are associated with separate servers.

29. (Original) The computer program product of claim 25, wherein the first and second components are Enterprise Java Bean components.

30. (Original) The computer program product of claim 29, wherein the first and second components are associated with a container.

31. (Currently Amended) A computer program product for an enterprise environment associated with a computing system, the computer program product comprising:

an asynchronous proxy for receiving a request from a first object-oriented component residing at a first server, the request intending to invoke a second object-oriented component residing at a second server wherein the request has a void return type and is not associated with application-specific exceptions;

an exception listener object-oriented component coupled to the asynchronous proxy and registered for the second object-oriented component, wherein the exception listener uses a scope corresponding to the request to handle exceptions associated with the invocation of the second object-oriented component, wherein the exception listener is stateless and is operable to handle a plurality of types of exceptions from a plurality of different components.

32. (Original) The computer program product of claim 31, wherein the request has a return type of void.

33. (Original) The computer program product of claim 31, wherein the request is associated with no application specific exceptions.

34 (Canceled) The computer program product of claim 31, wherein the first and second components are associated with separate servers.

35. (Original) The computer program product of claim 31, wherein the first and second components are Enterprise Java Bean components.

36. (Original) The computer program product of claim 35, wherein the first and second components are associated with a container.

37. (Original) The computer program product of claim 31, wherein the worker thread invokes the second components after receiving a transaction commit signal from the container.

38. (Currently Amended) A computer system operating an enterprise environment, the computer system comprising:

a processor coupled to memory;

an interface coupled to the processor, the interface configured to transmit a request from a first object-oriented component residing at the first server to invoke a ~~the~~ second object-oriented component residing at a second server, wherein the request has a void return type and is not associated with application-specific exceptions the first and second object-oriented components operating in environments allowing direct invocation of the second component by the first component, wherein the interface also transmits information to register an exception listener on an asynchronous proxy, the asynchronous proxy associated with the second component and the exception listener being registered for the second component and, wherein the exception listener is stateless and is operable to handle a plurality of types of exceptions from a plurality of different components.

39. (Currently Amended) The computer system ~~computer-implemented method~~ of claim 38, wherein the request has a return type of void.

40. (Currently Amended) The computer system ~~computer-implemented method~~ of claim 38, wherein the request is associated with no application specific exceptions.

41. (Cancelled) The computer system computer-implemented method of claim 38, wherein the first and second components are associated with separate servers.

42. (Currently Amended) The computer system ~~computer-implemented method~~ of claim 38, wherein the first and second components are Enterprise Java Bean components.

43. (Currently Amended) An apparatus for a first component to invoke a second component asynchronously in an object-oriented computing environment, the apparatus comprising:

means for receiving at an asynchronous proxy an asynchronous request from a first object-oriented component residing at a first server to invoke a second object-oriented component residing at a second server, wherein the request has a void return type and is not associated with application-specific exceptions ~~wherein the first server is connected to the second server over a network interface~~;

means for setting an exception listener on the asynchronous proxy and a scope of the second component, the exception listener being registered for the second component;

~~means for maintaining the scope of the received request at the second server;~~

means for storing the request and the scope in a queue on the asynchronous proxy;

means for providing a thread for identifying the received request and invoking the second component, wherein the thread identifies an exception listener object-oriented component for handling exceptions associated with the invocation of the second component, the exception listener object-oriented component registered on an asynchronous proxy, wherein the request is associated with no application specific exceptions.

44. (Canceled) An apparatus for a first component to invoke a second component asynchronously in an object-oriented environment, the computer-implemented method comprising:

transmitting a request from a first object-oriented component residing at a first server to invoke a second object-oriented component residing at a second server, the first and second object-oriented components operating in environments allowing direct invocation of the second component by the first component;

registering an exception listener object-oriented component on an asynchronous proxy, the asynchronous proxy associated with the second component, wherein the exception listener is stateless and is operable to handle a plurality of types of exceptions from a plurality of different components.